Systems Design Document

# Introduction

This document is a high-level, language-independent description of the system design. It is not intended to replace a low-level, detailed technical design document.

# Proposed Software Architecture

## Overview

The system will be comprised of two major parts: a server and a client. The server shall be responsible for all persistence and implement the game rules, while the client shall be responsible for the end-user look-and-feel and little else.

## Subsystem Descriptions

*[insert component diagram here]*

### Server

The server shall send and receive information via a strictly-defined JSON API. The server shall maintain the state of each game, as well as the state of the lobby. The server shall receive information about player actions from the clients and respond with the updated state of the game after that action was performed, or an error if the action is invalid. The server shall maintain user information for each player, including the master count of chips in the system. The server shall also be responsible for validation of game state and player actions.

### Client

The client shall present a user interface to the user that is clean and discoverable. The client shall allow the player to make their choices, and send the information to the server using the JSON API. The client shall update when the server pushes out a new game state. The client shall present information about the game state to the player.

## Persistent Data Management

The server shall maintain persistent data about users and their chip count in a SQL Lite database.

*[Insert entity-relationship diagram here]*

## Access Control and Security

### Password Selection

The user shall be able to select a username and a password upon registration, as well as an email address. The user shall be able to change their password at will. An email can be triggered to be sent to the user to reset their password if need be.

### Password Storage [not final]

The password shall be stored encrypted with the bcrypt hashing algorithm, and shall not be recoverable. A random salt shall be generated for each user and appended to their password before hashing; the salt shall be stored in the database.

### Authentication Token

Upon receiving a successful username and password combination, the server shall reply with a one-use token that will expire in 24 hours. The client shall send that token along with every subsequent message; if a message other than a login attempt is sent without a token, the server shall reject it.